EclipseCon France
2017-06-21

Xtending our VHDL Xtext formatter with the formatter2 API
ir. Titouan Vervack
EclipseCon France, Toulouse
2017-06-21

Xtending our VHDL Xtext formatter with the formatter2 API
ir. Titouan Vervack

redefine.digital.design: Helping you deal with complexity in VHDL and Verilog.
Ghent
What do we do?

VHDL & SV IDE

```vhdl
entity leds is
  port(
    clk : in std_logic;
    led1, led2, led3, led4, led5 : out std_logic
  );
end entity leds;

architecture blink of leds is
  signal clk_4hz : std_logic;
begins
  process(clk) is
    variable counter : unsigned(23 downto 0);
begins
    if rising_edge(clk) then
      if counter = 2_999_999 then
        counter := x'000000';
        clk_4hz <= not clk_4hz;
      else
        counter := counter + 1;
      end if;
    end if;
  end process;

  led1 <= clk_4hz;
  led2 <= clk_4hz;
  led3 <= clk_4hz;
  led4 <= clk_4hz;
  led5 <= not clk_4hz;
end architecture blink;
```
What do we do?

Advanced analysis and linting

```vhdl
-- unused ports
-- naming conventions
-- signals/variables read/write/use
-- dead code
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ent is
  port (  
    clk : in std_logic;
    rst : in std_logic;
    input : in integer;
    output : out integer
  );
end entity ent;

architecture RTL of ent is
  signal not_conventional : integer;
  signal s_conventional : integer;
begin
  s_conventional <= 25 + input;
  not_conventional <= 15 + s_conventional;
  output <= not_conventional / 2;
  name : process (clk) is
    variable only_read : integer := 42;
    variable only_written : std_logic := 'X';
    begin
      if rising_edge(clk) then
        if input = only_read then
          only_written := 'H';
        elsif 15 > work.constants.ANSWER then
          only_written := 'L';
        end if;
      end if;
    end process name;
end architecture RTL;
```
What do we do?

Visualisation
What do we do?

Visualisation
What do we do?

Eat cake
What do we do?

And cookies...
What do we do?

And occasionally... formatting

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

-- @formatter:off
entity myfoobar is port(clk : in std_logic); end entity myfoobar;
-- @formatter:on

LIBRARY ieee;
USE ieee.std_logic_1164.all;

entity foobar is
  port(  clk : in std_logic; -- This is the clock
        rst : in std_logic; -- This is the reset
        longsignal : inout std_logic_vector -- This is a long signal
  );
end entity foobar;

-- Below follows the architecture
architecture RTL of foobar is
  signal asignal : std_logic;
signal intsignal : integer;
signal wire : std_logic; -- I like my types under the name
begin
end architecture RTL;
```
What do we do?

And occasionally... formatting

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

-- @formatter:off
entity myfoobar is port(clk : in std_logic); end entity myfoobar;
-- @formatter:on

LIBRARY ieee;
USE ieee.std_logic_1164.all;
entity foobar is
  port( clk : in std_logic; -- This is the clock
       rst : in std_logic; -- This is the reset
       longsignal : inout std_logic_vector -- This is a long signal
     );
end entity foobar;

-- Below follows the architecture
architecture RTL of foobar is
  signal asignal : std_logic;
  signal intsignal : integer;
  signal wire : std_logic; -- I like my types under the name
begin
  end architecture RTL;
end;
```
Our formatter problems

- Big grammar
  - ~1.2k LoC
  - ~200 rules
Our formatter problems

- Big grammar
  - ~1.2k LoC
  - ~200 rules

- Hard to fix bugs
Our formatter problems

● Big grammar
  ○ ~1.2k LoC
  ○ ~200 rules

● Hard to fix bugs

● Near impossible to add new features
## Formatter 1.0 vs Formatter 2.0

<table>
<thead>
<tr>
<th>Formatter 1.0</th>
<th>Formatter 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No access to</strong></td>
<td><strong>Access to</strong></td>
</tr>
<tr>
<td>– Node model</td>
<td>– Node model</td>
</tr>
<tr>
<td>– AST</td>
<td>– AST</td>
</tr>
<tr>
<td>– Text</td>
<td>– Grammar</td>
</tr>
<tr>
<td>=&gt; Specify everything purely on the grammar</td>
<td>=&gt; Context aware formatting</td>
</tr>
<tr>
<td>=&gt; Lots of hacks</td>
<td>– Conditional formatting</td>
</tr>
<tr>
<td></td>
<td>– Table formatting</td>
</tr>
<tr>
<td><strong>Can’t take context into account</strong></td>
<td><strong>Very customizable</strong></td>
</tr>
<tr>
<td>– Can’t adjust to the user</td>
<td></td>
</tr>
<tr>
<td><strong>Not customizable</strong></td>
<td></td>
</tr>
</tbody>
</table>

Note: This comparison highlights the differences between Formatter 1.0 and Formatter 2.0 in terms of access, context awareness, and customization.
Formatter 2.0 region example

variable foo : bit; -- bar

https://de.slideshare.net/meysholdt/xtexts-new-formatter-api
Formatting 2.0 usage example

https://de.slideshare.net/meysholdt/xtexts-new-formatter-api
class Awesom04000 extends AbstractFormatter2 {

}
class AwesomO4000 extends AbstractFormatter2 {
    def dispatch void format(FunctionDecl fd, extension IFormattableDocument d) {
    }
}

https://de.slideshare.net/meysholdt/xtexts-new-formatter-api
class AwesomO4000 extends AbstractFormatter2 {

    def dispatch void format(FunctionDecl fd, extension IFormattableDocument d) {
        interior(fd.regionFor.keyword("{"), fd.regionFor.keyword("}"))[indent]

    }

}
```java
class AwesomO4000 extends AbstractFormatter2 {

    def dispatch void format(FunctionDecl fd, extension IFormattableDocument d) {
        interior(fd.regionFor.keyword("{"), fd.regionFor.keyword("}"))
        fd.declarations.forEach[format]
    }
}

https://de.slideshare.net/meysholdt/xtexts-new-formatter-api
```
class AwesomO4000 extends AbstractFormatter2 {

    def dispatch void format(FunctionDecl fd, extension IFormattableObject d) {
        interior(fd.regionFor.keyword("{"), fd.regionFor.keyword("")}[indent]

        fd.declarations.forEach[format]

        fd.regionFor.keyword(";")
    }
}

https://de.slideshare.net/meysholdt/xtexts-new-formatter-api
class AwesomO4000 extends AbstractFormatter2 {

    def dispatch void format(FunctionDecl fd, extension IFormattableDocument d) {
        interior(fd.regionFor.keyword("{")), fd.regionFor.keyword("}"))[indent]

        fd.declarations.forEach[format]

        fd.regionFor.keyword(";")
        .prepend[noSpace highPriority]
    }
}
class AwesomO4000 extends AbstractFormatter2 {

def dispatch void format(FunctionDecl fd, extension IFormattableDocument d) {
    interior(fd.regionFor.keyword("{"), fd.regionFor.keyword(""}"))

    fd.declarations.forEach[format]

    fd.regionFor.keyword(";")
        .prepend[noSpace highPriority]
        .append[setNewlines(1, 1, 2)]
}
}

https://de.slideshare.net/meysholdt/xtexts-new-formatter-api
Time for a plan
Time for a plan

1. Make tests compatible
Time for a plan

1. Make tests compatible
2. Start out easy: only indentation
Time for a plan

1. Make tests compatible
2. Start out easy: only indentation
3. Fix tests one by one
Time for a plan

1. Make tests compatible
2. Start out easy: only indentation
3. Fix tests one by one
4. Fix old bugs
Time for a plan

1. Make tests compatible
2. Start out easy: only indentation
3. Fix tests one by one
4. Fix old bugs
5. Do not introduce new bugs
Time for a plan

1. Make tests compatible
2. Start out easy: only indentation
3. Fix tests one by one
4. Fix old bugs
5. Do not introduce new bugs
6. Feature parity & extend with new features
1. Make tests compatible
2. Start out easy: only indentation
3. Fix tests one by one
4. Fix old bugs
5. Do not introduce new bugs
6. Feature parity & extend with new features
7. Performance tweaking to same or better level than old
1. Make tests compatible

```python
formatter.format(node, node.offset, node.length).formattedText
```
1. Make tests compatible

```java
formatter.format(node, node.length).formattedText
```
1. Make tests compatible

```kotlin
val request = new FormatterRequest()
val parsed = new XtextResource()
parsed.contents.add(parseResult.rootASTElement)
parsed.setParseResult(parseResult)
val access = builder.nodeNameModel(resource).create()
request.setTextRegionAccess(access)
val replacements = formatter.format(request)
access.rewriter.renderToString(replacements)
```
2. Start out easy: only indentation

\texttt{interior(entity.regionFor.keyword(“is”), entity.regionFor.keyword(“end”))}\texttt{[indent]}
2. Start out easy: only indentation

interior(entity.regionFor.keyword("is"), entity.regionFor.keyword("end"))[indent]

- Indentation requires newline definitions
2. Start out easy: only indentation

```
interior(entity.regionFor.keyword("is"), entity.regionFor.keyword("end"))[indent]
```

- Indentation requires newline definitions

- => replace all newlines with... the same amount of newlines
Actual timeline

3. Fix tests one by one
4. Fix old bugs
Actual timeline

3. Fix tests one by one
4. Fix old bugs
Actual timeline

5. Do not introduce new bugs
Actual timeline

5. Do not introduce new bugs
6. Feature parity & extend formatter with new features
6. Feature parity & extend formatter with new features

- Parity:
6. Feature parity & extend formatter with new features

- Parity:
  - Keyword casing
6. Feature parity & extend formatter with new features

- Parity:
  - Keyword casing
  - Comment alignment to columns
6. Feature parity & extend formatter with new features

- Parity:
  - Keyword casing
  - Comment alignment to columns
  - Vertical alignment on keywords “:”, “:=”,...
6. Feature parity & extend formatter with new features

- Parity:
  - Keyword casing
  - Comment alignment to columns
  - Vertical alignment on keywords 
    
  - Preserve newline
6. Feature parity & extend formatter with new features

- Parity:
  - Keyword casing
  - Comment alignment to columns
  - Vertical alignment on keywords “:”, “:=”,...
  - Preserve newline

- New features:
6. Feature parity & extend formatter with new features

- **Parity:**
  - Keyword casing
  - Comment alignment to columns
  - Vertical alignment on keywords “:”, “:=”,...
  - Preserve newline

- **New features:**
  - Correct indentation
6. Feature parity & extend formatter with new features

- **Parity:**
  - Keyword casing
  - Comment alignment to columns
  - Vertical alignment on keywords “:”, “:=”,...
  - Preserve newline

- **New features:**
  - Correct indentation
  - Formatter tags
6. Feature parity & extend formatter with new features

● Parity:
  ○ Keyword casing
  ○ Comment alignment to columns
  ○ Vertical alignment on keywords ":", "=",...
  ○ Preserve newline

● New features:
  ○ Correct indentation
  ○ Formatter tags

Easy once we have alignment
Alignment?

```vhdl
port(clk : in std_logic := 'X';
    rst : in std_logic := 'X';
    longsignal : inout std_logic_vector := ('0',
                                                '1')
    );
```
```vhdl
port(clk : in  std_logic := 'X';
    rst : in  std_logic := 'X';
    longsignal : inout std_logic_vector := ('0', '1'))
```
- Replacements are added in random order, applied in correct order
Alignment?

- Replacements are added in random order, applied in correct order
- Alignment needs to be done on post-formatted input
● Replacements are added in random order, applied in correct order

● Alignment needs to be done on post-formatted input

● How do I know which region to format?
port(clk : in std_logic := 'X';
    rst : in   std_logic := 'X';
    longsignal : inout std_logic_vector := ('0', '1'));

port(clk : in std_logic := 'X';
    rst : in   std_logic := 'X';
    longsignal : inout std_logic_vector := ('0', '1'));
port(clk : in std_logic := 'X';
    rst : in std_logic := 'X';
    longsignal : inout std_logic_vector := ('0', '1'));

port(clk : in std_logic := 'X';
    rst : in std_logic := 'X';
    longsignal : inout std_logic_vector := ('0', '1'));}
align the code to improve readability
port(  clk : in  std_logic := 'X';
       rst : in  std_logic := 'X';
       longsignal : inout std_logic_vector := ('0', '1')
     );
port(  clk : in  std_logic := 'X';
    rst: in    std_logic := 'X';
  longsignal : inout std_logic_vector := ('0', '1');
);

start

end
port(clk : in std_logic := 'X';
rst : in std_logic := 'X';
longsignal : inout std_logic_vector := ('0', '1');
);

port(clk : in std_logic := 'X';
rst : in std_logic := 'X';
longsignal : inout std_logic_vector := ('0', '1');
);

Alignment!
Find smallest encapsulating object that formats region between start and end
Find smallest encapsulating object that formats region between start and end
clk : in std_logic := 'X';
clk : in std_logic := 'X';

start

end
clk : in std_logic := 'X';

```java
PortDeclaration
  start
  PortList
  PortDeclaration
    name
    direction
    type
    value
  end
  end.getSemanticElement()
end
```
clk : in std_logic := 'X';

end.getSemanticElement()
Alignment!

Format smallest encapsulating object

```scala
val rule = portDecl.getParserRule()
val oldAccess = createTextRegionAccess(rule, portDecl.text)
val formatted = portDecl.format()
```
val rule = portDecl.getParserRule()
val oldAccess = createTextRegionAccess(rule, portDecl.text)
val formatted = portDecl.format()

val newAccess = createTextRegionAccess(rule, formatted.text)
val oldIndices = oldAccess.getIndices(start, end)
val length = newAccess.countBetween(oldIndices)

Format smallest encapsulating object

Map unformatted string to formatted string
→return ( foo((5, 6), ...3));
return foo((5, 6), (3));
Align on first parameter?

```c
return (foo((5, 6), 3));
```

```c
return (5, foo((5, 6), 3));
```
→ return ( foo((5, 6), 3));

→ return (5, foo((5, 6), 3));
Align on first parameter?

```python
return foo((5, 6), (3));
```
Align on first parameter?

\[
\text{return } \left( \text{foo}(5, \ldots \cdot 6), \ldots \cdot 3 \right); \\
\text{return } \left( \ldots \cdot \text{foo}(5, \ldots \cdot 6), \ldots \cdot 3 \right); \\
\text{return } \left( 5, \ldots \cdot \text{foo}(5, \ldots \cdot 6), \ldots \cdot 3 \right); \\
\text{return } \left( 5, \ldots \cdot \text{foo}(5, \ldots \cdot 6), \ldots \cdot 3 \right); \\
\]
Align on first parameter?

\[
\text{return } (\text{foo}((5, 6), 3));
\]

\[
\text{return } \text{foo}(5, \text{foo}(6, 3));
\]

\[
\text{return } (5, \text{foo}(6, 3));
\]

\[
\text{return } \text{foo}(5, (6, 3));
\]
Align on first parameter?

\[
\text{return } (\text{foo}((5, 6), \cdot 3));
\]

\[
\text{return } (5, \text{foo}((5, 6), \cdot 3));
\]

\[
\text{return } \text{foo}((5, 6), \cdot 3));
\]

\[
\text{return } \text{foo}((5, 6), \cdot 3));
\]
Align on first parameter?

\[
\text{return } (\text{foo}(5, 6), 3) \text{;}
\]

Find largest encapsulating element on same line

\[
\text{return } (5, \text{foo}(5, 6), 3) \text{;}
\]

\[
\text{return } \text{foo}(5, 6, 3) \text{;}
\]

\[
\text{return } \text{foo}(5, 6, 3) \text{;}
\]
Actual timeline

6. Feature parity & extend formatter with new features

Correct indentation
Actual timeline

6. Feature parity & extend formatter with new features

Correct indentation

```scala
val replacements = formatter.format(request)
```
Actual timeline

6. Feature parity & extend formatter with new features

Correct indentation

```java
val replacements = formatter.format(request)
replacements
    .filter[ r|
```
6. Feature parity & extend formatter with new features

Correct indentation

```scala
val replacements = formatter.format(request)
replacements
  .filter[r | r.lineCount > 1
```

```
6. Feature parity & extend formatter with new features

Correct indentation

```scala
val replacements = formatter.format(request)
replacements
  .filter[ r|
    r.lineCount > 1 &&
    r.replacementText.newlines + 1 == r.lineCount
  ]
```
6. Feature parity & extend formatter with new features

Formatter tags
6. Feature parity & extend formatter with new features

Formatter tags

-- @formatter:off Turns formatter off
-- @formatter:on  Turns formatter on
Actual timeline

6. Feature parity & extend formatter with new features

Formatter tags

-- @formatter:off  Turns formatter off
-- @formatter:on   Turns formatter on

```
val originalRegions = request.getRegions() ?: document.region
```
6. Feature parity & extend formatter with new features

Formatter tags

    -- @formatter:off  Turns formatter off
    -- @formatter:on   Turns formatter on

```kotlin
val originalRegions = request.getRegions() ?: document.region
val betweenRegions = document.findRegionsBetween(off, on)
```
Actual timeline

6. Feature parity & extend formatter with new features

Formatter tags

```
-- @formatter:off Turns formatter off
-- @formatter:on  Turns formatter on
```

```kotlin
val originalRegions = request.getRegions() ?: document.region
val betweenRegions = document.findRegionsBetween(off, on)
.addAll(document.findRegionsBetween(off, EOF))
```
Actual timeline

6. Feature parity & extend formatter with new features

Formatter tags

```kotlin
val originalRegions = request.getRegions() ?: document.region
val betweenRegions = document.findRegionsBetween(off, on)
    .addAll(document.findRegionsBetween(off, EOF))
val newRegions = originalRegions.removeRegions(betweenRegions)
```
6. Feature parity & extend formatter with new features

Formatter tags

-- @formatter:off Turns formatter off
-- @formatter:on Turns formatter on

```kotlin
val originalRegions = request.getRegions() ?: document.region
val betweenRegions = document.findRegionsBetween(off, on)
    .addAll(document.findRegionsBetween(off, EOF))
val newRegions = originalRegions.removeRegions(betweenRegions)
request.setRegions(newRegions)
```
7. Performance tweaking to same or better level than old
7. Performance tweaking to same or better level than old
   - Cache ITextRegionAccess in subformats
7. Performance tweaking to same or better level than old
   
   - Cache ITextRegionAccess in subformats
   
   - Cache IParser.createAllRules(Grammar)
Performance

7. Performance tweaking to same or better level than old

- Cache ITextRegionAccess in subformats
- Cache IParser.createAllRules(Grammar)
- Merge all replacements into one big replacement
Performance

7. Performance tweaking to same or better level than old

![Formatter performance chart]

- Old
- New No Opt
- New W Cache No Merge
- New
- Emacs
- Xtend 2.12
- Java

LoC vs. Time (ms)
7. Performance tweaking to same or better level than old
Biggest problems in hindsight
Biggest problems in hindsight

● Tests pass, close enough?
Biggest problems in hindsight

- Tests pass, close enough?
  - Nope, old testset isn’t nearly enough
Biggest problems in hindsight

- Tests pass, close enough?
  - Nope, old testset isn’t nearly enough

- Comments are an enormous pain: eat newlines, add spaces, eat spaces, fake empty lines
Biggest problems in hindsight

- Tests pass, close enough?
  - Nope, old testset isn’t nearly enough

- Comments are an enormous pain: eat newlines, add spaces, eat spaces, fake empty lines

- Alignment interactions with normal and first param alignments and comments
Biggest problems in hindsight

● Tests pass, close enough?
  ○ Nope, old testset isn’t nearly enough

● Comments are an enormous pain: eat newlines, add spaces, eat spaces, fake empty lines

● Alignment interactions with normal and first parameter comments
Biggest problems in hindsight

- Tests pass, close enough?
  - Nope, old testset isn’t nearly enough

- Comments are an enormous pain: eat newlines, add spaces, eat spaces, fake empty lines

```vhdl
port(
    clk : in  std_logic_vector := ('0', '1');
    rst : in  bit_vector := ('0', '1')
); -- test

entity duttt is
```
Biggest problems in hindsight

- Tests pass, close enough?
  - Nope, old testset isn’t nearly enough

```vhdl
generate_label : -- Space on NL
_if if_label : true generate
begin

port(
  clk : in std_logic_vector := ('0', '1');
  rst : in bit_vector    := ('0', '1');
-- test
);
/*
 * test
 */
_entity duttt is
```
Biggest problems in hindsight

- Tests pass, close enough?
  - Nope, old testset isn't nearly enough
- Comments are an enormous pain: eat newlines, add spaces, eat spaces, fake empty lines
- Alignment interactions with normal and first param alignments and comments
Biggest problems in hindsight

- Tests pass, close enough?
  - Nope, old testset isn't nearly enough
- Comments are an enormous pain: eat newlines, add spaces, eat spaces, fake empty lines
- Alignment interactions with normal and first param alignments and comments

```vhdl
constant c cr array : t cr array(2**12 downto 0) := (16#00# => (others => '0'),
  -- Length
  port(
    16#02# => clk : in std_logic_vector := ('0', '1');
    16#03# => rst : in bit_vector := ('0',
      '1')
  );
  16#05# => );
  16#06# => x"01",

-- CR/CSR Data Access Width
-- CR/CSR Space Specification ID

generate_label : -- Space on NL _if if_label : true generate begin

port(
  clk : in std_logic_vector := ('0', '1');
  rst : in bit_vector := ('0', '1')
-- test
);

/* test */
_entity duttt is
```
Biggest problems in hindsight

- Tests pass, close enough?
  - Nope, old testset isn’t nearly enough
- Comments are an enormous pain: eat newlines, add spaces, eat spaces, fake empty lines
- Alignment interactions with normal and first param alignments and comments
Warning to Xtext users

Default ExceptionHandler logs DSL source code to stderr

```java
@org.eclipse.xtext.formatting2.internal.ConflictingRegionsException: Conflicting Regions.

16#1a1# -> x"00",
16#1a2# -> x"00",
16#1a3# -> x"00",
others ->(((others -> '0')))

```

```java
at org.eclipse.xtext.formatting2.internal.TextSegmentSet.handleConflict(TextSegmentSet.java:83)
at org.eclipse.xtext.formatting2.internal.TextSegmentSet.replaceExistingEntry(TextSegmentSet.java:139)
at org.eclipse.xtext.formatting2.internal.TextSegmentSet.add(TextSegmentSet.java:60)
at org.eclipse.xtext.formatting2.internal.FormattableObjectFormatParameterList.prepend(FormattableObjectFormatParameterList.java:26)
at com.sigasi.hdt.vhdl.formatting2.VhdlBasicFormatter.formatParameterList(VhdlBasicFormatter.java:2456)
at com.sigasi.hdt.vhdl.formatting2.VhdlBasicFormatter.formatVhdlBasicParameterList(VhdlBasicFormatter.java:2444)
at com.sigasi.hdt.vhdl.formatting2.VhdlBasicFormatter._format(VhdlBasicFormatter.java:2430)
at com.sigasi.hdt.vhdl.formatting2.VhdlBasicFormatter._format(VhdlBasicFormatter.java:3496)
at com.sigasi.hdt.vhdl.formatting2.VhdlBasicFormatter._format(VhdlBasicFormatter.java:194)
at org.eclipse.xtext.formatting2.VhdlBasicFormatter._format(VhdlBasicFormatter.java:5508)
at org.eclipse.xtext.formatting2.VhdlBasicFormatter._format(VhdlBasicFormatter.java:494)
at org.eclipse.xtext.formatting2.VhdlBasicFormatter._format(VhdlBasicFormatter.java:194)
at com.sigasi.hdt.vhdl.formatting2.VhdlBasicFormatter.lambda$273(VhdlBasicFormatter.java:2533)
at java.lang.Iterable.forEach(Iterable.java:75)
at com.sigasi.hdt.vhdl.formatting2.VhdlBasicFormatter._format(VhdlBasicFormatter.java:2535)
at com.sigasi.hdt.vhdl.formatting2.VhdlBasicFormatter._format(VhdlBasicFormatter.java:3496)
at org.eclipse.xtext.formatting2.VhdlBasicFormatter._format(VhdlBasicFormatter.java:194)
at org.eclipse.xtext.formatting2.AbstractFormatter2._format(AbstractFormatter2.java:263)
```
Sigasi future work
Sigasi future work

- Smooth out remaining bugs
Sigasi future work

- Smooth out remaining bugs
- Some more context awareness
Sigasi future work

- Smooth out remaining bugs
- Some more context awareness
- Autowrap
Sigasi future work

- Smooth out remaining bugs
- Some more context awareness
- Autowrap
- Performance
Sigasi future work

- Smooth out remaining bugs
- Some more context awareness
- Autowrap
- Performance
- Upstream features
Sigasi future work

- Smooth out remaining bugs
- Some more context awareness
- Autowrap
- Performance
- Upstream features
  - Formatter tags
Sigasi future work

- Smooth out remaining bugs
- Some more context awareness
- Autowrap
- Performance
- Upstream features
  - Formatter tags
  - Correct indentation
Sigasi future work

- Smooth out remaining bugs
- Some more context awareness
- Autowrap
- Performance
- Upstream features
  - Formatter tags
  - Correct indentation
  - Performance optimizations
Xtext wish list
Accept my PRs (tags, correct indent, performance)
Xtext wish list

- Accept my PRs (tags, correct indent, performance)
- Fix/discuss my issues
Xtext wish list

- Accept my PRs (tags, correct indent, performance)
- Fix/discuss my issues
- Fix Xtend formatter issues
Xtext wish list

- Accept my PRs (tags, correct indent, performance)
- Fix/discuss my issues
- Fix Xtend formatter issues
- Very slow Xtend formatter source file
Xtext wish list

- Accept my PRs (tags, correct indent, performance)
- Fix/discuss my issues
- Fix Xtend formatter issues
- Very slow Xtend formatter source file
- Add pre- and post format hooks in format(FormatterRequest)
Conclusion

● Context aware
Conclusion

- Context aware
- Debugable
Conclusion

● Context aware
● Debugable
● Good performance
Conclusion

- Context aware
- Debugable
- Good performance
- Very customizable
Conclusion

- Context aware
- Debugable
- Good performance
- Very customizable
- User friendly
Conclusion

- Context aware
- Debugable
- Good performance
- Very customizable
- User friendly
  - Done in 2 months (base in 2 weeks) while first task and not used to Xtend
Conclusion

- Context aware
- Debugable
- Good performance
- Very customizable
- User friendly
  - Done in 2 months (base in 2 weeks) while first task and not used to Xtend
- Happy with the switch
● Context aware
● Debugable
● Good performance
● Very customizable
● User friendly
  ○ Done in 2 months (base in 2 weeks) while first task and not used to Xtend
● Happy with the switch
  ○ New features
Conclusion

- Context aware
- Debugable
- Good performance
- Very customizable
- User friendly
  - Done in 2 months (base in 2 weeks) while first task and not used to Xtend
- Happy with the switch
  - New features
  - Very readable
Conclusion

- Context aware
- Debugable
- Good performance
- Very customizable
- User friendly
  - Done in 2 months (base in 2 weeks) while first task and not used to Xtend
- Happy with the switch
  - New features
  - Very readable
  - Conditional formatting
Evaluate the Sessions

| -1 | 0 | +1 |

Sign in and vote at eclipsecon.org